Куча
Строки

Д. В. Луцив

Кафедра системого программирования СПбГУ

CS103

Содержание

1. Куча
   - Блоки одинаковой длины
   - Метод подходящих
   - Метод граничных маркеров
   - Метод двоичных близнецов
2. Строки
   - Изменяемые
   - Неизменяемые
   - Списки
   - Деревья

Суть

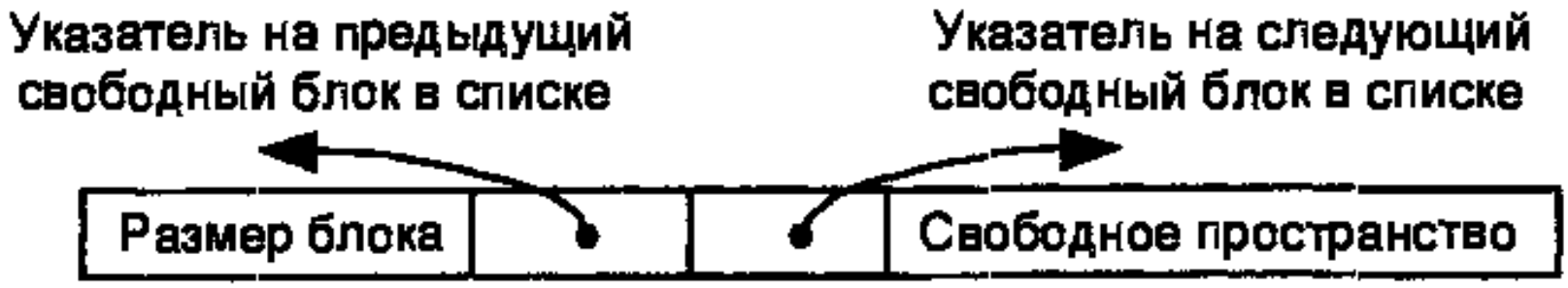
- Свободный блок хранит указатель на следующий свободный блок
- Свобоные блоки образуют стек на базе односвязного списка
- Все операции за $O(1)$

Суть

- Свободный блок хранит указатель на следующий свободный блок
- Свобоные блоки образуют стек на базе односвязного списка
- Все операции за $O(1)$

Суть

- Список м.б. двусвязным и циклическим;
- в свободном блоке хранится длина и указатели на следующий и предыдущие свободные, в занятом — только длину;
- при освобождении блоки надо сливать, дефрагментируя кучу.



а) Структура свободного блока памяти

б) Структура выделенного блока памяти

Недостаток

При возвращении блока в список свободных часто некогда искать, в какое место его поставить. Т.е. можно либо медленно освобождать память, сохраняя упорядоченность списка по адресам, либо освобождать быстро, но медленно дефрагментировать, бегая по неупорядоченному списку в поисках соседних блоков.

Вариации

- Метод первого подходящего
- Метод следующего подходящего

Суть

Позволяет оптимизировать дефрагментацию. Для этого на концах блока памяти, свободного или занятого, записываются маркеры, показывающие его состояние. Кроме того, на *обоих* концах свободного блока записывается его длина. Это позволяет быстро выяснить, заняты ли следующий и предыдущий блоки и произвести возможные слияния. У выделенного блока длина хранится только в начале.

Такая система позволяет, вообще говоря, совсем отказаться от списка, т.к. из каждого блока можно получить его размер, а значит и адрес следующего, и, более того, если предыдущий свободен (иначе не нужно), то и адрес предыдущего.

Суть

Позволяет оптимизировать дефрагментацию. Для этого на концах блока памяти, свободного или занятого, записываются маркеры, показывающие его состояние. Кроме того, на *обоих* концах свободного блока записывается его длина. Это позволяет быстро выяснить, заняты ли следующий и предыдущий блоки и произвести возможные слияния. У выделенного блока длина хранится только в начале.
Такая система позволяет, вообще говоря, совсем отказаться от списка, т.к. из каждого блока можно получить его размер, а значит и адрес следующего, и, более того, если предыдущий свободен (иначе не нужно), то и адрес предыдущего.

а) Структура свободного блока памяти

б) Структура выделенного блока памяти

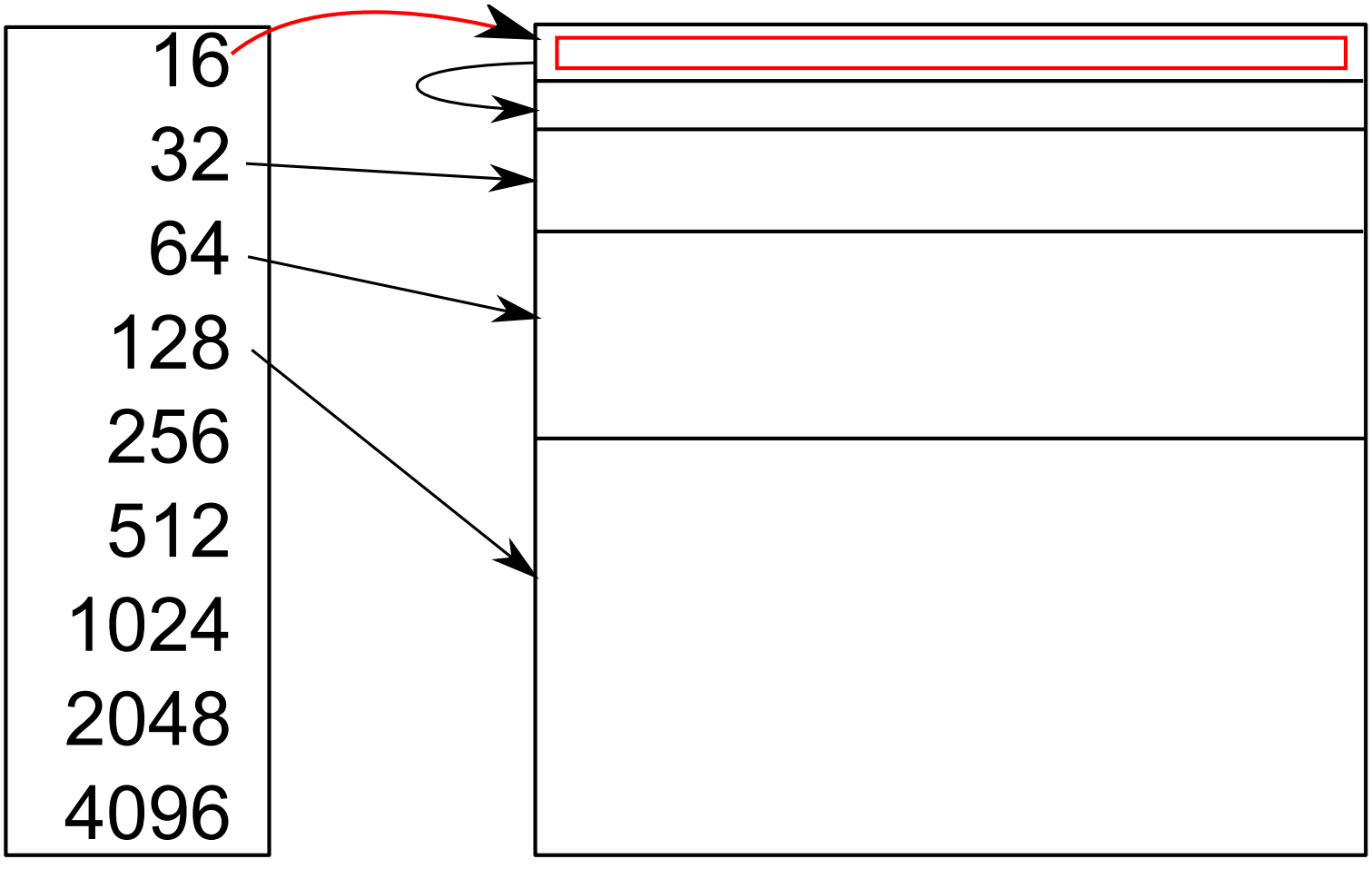
Недостаток

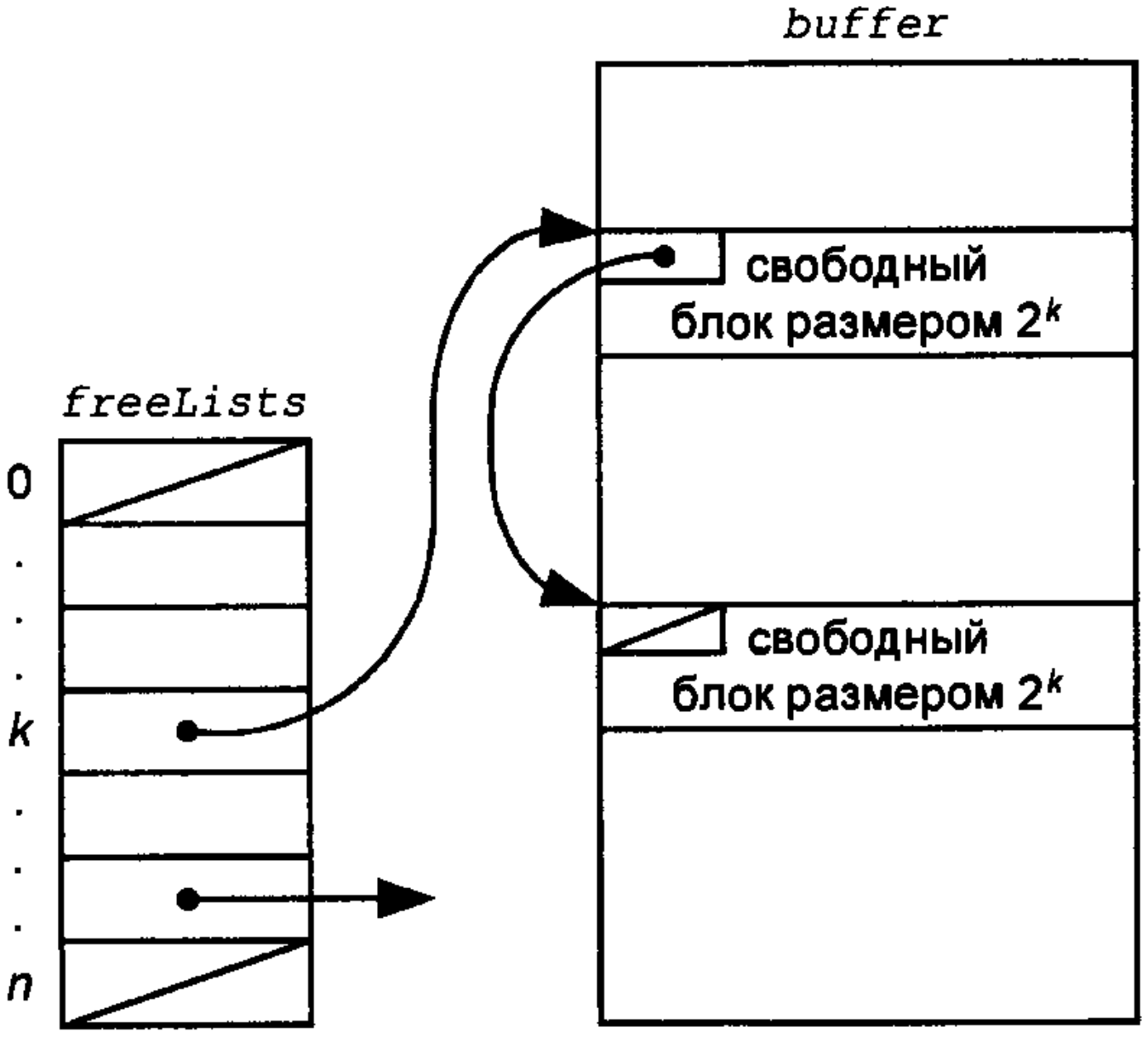
Накладные расходы по памяти. Нужно место под:

1. маркеры;
2. для свободного блока – под две копии длины, что не страшно, т.к. у занятого тоже одна копия хранится, т.е. можно выделить `sizeof(void *)` памяти.

Выделение

- Позволяет распределять память только блоками по 2^k – `sizeof(char)` ячеек.
- Заводится массив, k-й элемент которого хранит указатель на первый свободный элемент размером 2^k. Свободный элемент хранит указатель на следующий.
- Если блок нужного объема есть, то он выделяется и убирается из списка. Если нет, то выделяется (рекурсивно) блок в два раза больший, разбивается пополам, половина предоставляется программе, а её близнец отправляется в список свободных. Рекурсия может дойти доверху.


16
32
64
128
256
512
1024
2048
4096


buffer
свободный
блок размером 2^k
свободный
блок размером 2^k
freeLists
0
k
n

Освобождение и дефрагментация

- Каждый блок размером 2^k начинается с адреса, кратного 2^k (если считать от начала кучи).
- Таким образом можно по адресу и размеру любого блока найти его «близнеца», определив, является ли наш блок первым или вторым.

Близнец хранится в списке свободных блоков данного размера?

- Нет. Добавляем освобождаемый блок в список и ничего больше не делаем.
- Да. Убираем близнеца из списка свободных и рекурсивно пытаемся освободить уже вдвое больший блок, состоящий из исходного блока и его близнеца.

Освобождение и дефрагментация

- Каждый блок размером 2^k начинается с адреса, кратного 2^k (если считать от начала кучи).
- Таким образом можно по адресу и размеру любого блока найти его «близнеца», определив, является ли наш блок первым или вторым.

Близнец хранится в списке свободных блоков данного размера?

- Нет. Добавляем освобождаемый блок в список и ничего больше не делаем.
- Да. Убираем близнеца из списка свободных и рекурсивно пытаемся освободить уже вдвое больший блок, состоящий из исходного блока и его близнеца.

Благодарность

Использованы иллюстрации из книги:
Кубенский А. А. Структуры и алгоритмы обработки данных.
Объектно-ориентированный подход и реализация на C++.

Обычно представляются массивами

- LASCII
  - встроенные в Turbo Pascal (макс. размен 255 фиксирован)
  - короткие в Delphi
  - C++
  - иногда короткие локально, а длинные в куче
- ASCIIZ
  - C

Где и как

- Используются многими современными объектно - ориентированными системами, такими как .NET, Java, Python и т.д.
- Представляют собой объекты, инкапсулирующие массив символов, но не дающие его изменять.

Преимущества и недостатки

- Очень быстро копируются, т.к. копируются только ссылки, одинаковые строки могут храниться в одной и той же памяти.
- При любом изменении строки приходится создавать новый объект с новой строкой.
- Для конкатенации длинных строк принято использовать специальные объекты класса `StringBuffer` (Java) или `StringBuilder` (.NET), реализующие изменяемые строки.

Экономия

- Разные объекты строкового тип могут ссылаться на фрагменты массивов (указывать в середину и хранить небольшое значение длины). Это требует аккуратного управления выделением памяти и возможности программирования на уровне C.
- .NET иногда делает подобные оптимизации.

Haskell

- Ленивость потенциально позволяет работать с бесконечными строками, например, с потоками текста.
- Преимущества неизменяемых строк, в т.ч. хвост строки без копирования.
- Скорость доступа к отдельным символам сильно зависит от того, как система оптимизирует работу со списками.
- Конкатенация совсем не быстрая.

Rope

Одна из самых популярных реализаций – Rope – предложена Hans Boehm, HP (автор одного из самых популярных консервативных сборщиков мусора для C) в 1980 годах.

Подход

- Строка представляется при помощи бинарного дерева.
- В каждой вершине ссылки на потомков или на память с символами (или на другую реализацию строк). Во внутренних вершинах вместе со ссылками хранят длины строк, представленных левым и правым детьми (чтобы не пересчитывать).
- Структура неизменяемая, фрагменты дерева можно повторно использовать (получается направленный ациклический граф).

Операции

- Конкатенация мгновенная, создается новая вершина, указывающая на строку и добавку.
- Балансировка происходит путем вращения. Цель – не наименьшая высота, а примерно одинаковая длина строк в поддеревьях.

Конкатенация

- При конкатенации длинной строки и короткой добавки, однако, дерево получается неоптимальным.
- Каждый раз перестраивать такое дерево невыгодно.
- Но можно, если пользуемся счетчиком ссылок (имеем право, т.к. циклов нет) и уверены, что ссылка единственная, не создать новое дерево, а перестроить, используя старые вершины. Если система программирования позволяет, конечно. Haskell, например, не позволит.

Занятно: при генерации текста высокоурвневой программы по её синтаксическому дереву полученный Rope (без балансировки) будет изоморфен дереву программы.

Конкатенация

- При конкатенации длинной строки и короткой добавки, однако, дерево получается неоптимальным.
- Каждый раз перестраивать такое дерево невыгодно.
- Но можно, если пользуемся счетчиком ссылок (имеем право, т.к. циклов нет) и уверены, что ссылка единственная, не создать новое дерево, а перестроить, используя старые вершины. Если система программирования позволяет, конечно. Haskell, например, не позволит.

Занятно: при генерации текста высокоурвневой программы по её синтаксическому дереву полученный Rope (без балансировки) будет изоморфен дереву программы.

Вопросы

EDU.DLUCIV.NAME